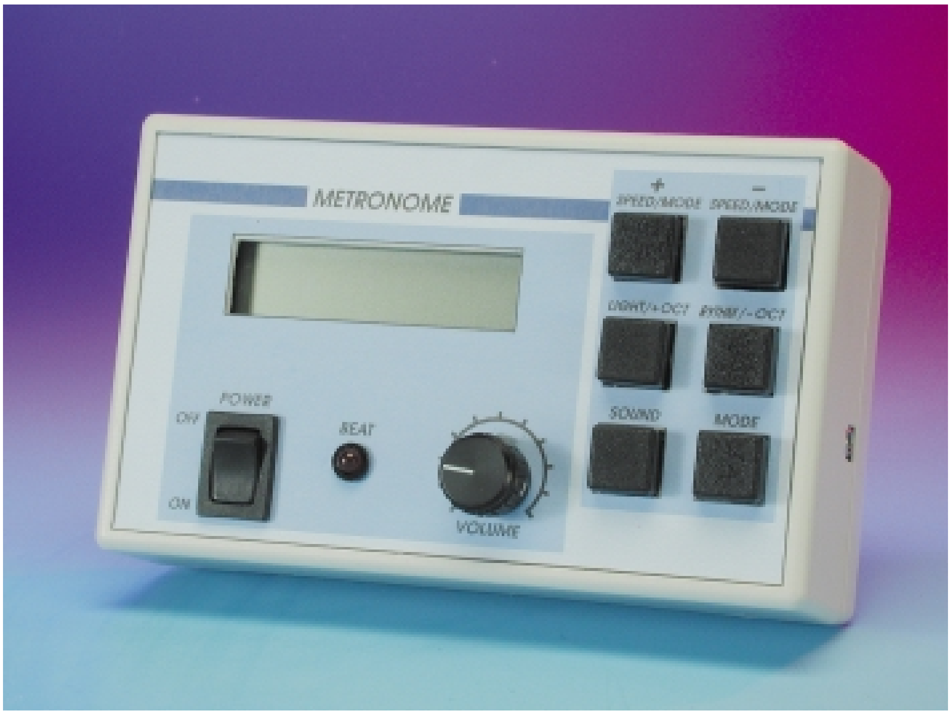
Metronome/Diapason

based on a PIC16F84 microcontroller

By C. Salmeron

This article describes a metronome and a diapason (electronic tuning fork) combined into a single unit. The circuit design philosophy was to keep the hardware as simple as possible, and to provide features comparable to those of high-end commercial units. This has been achieved by using a microcontroller chip PIC16F84.



The PIC16C84 was programmed to take care of all functions, including the generation of beat *ticks* in metronome mode, the digital synthesis of pure tones (sinewaves) for the full range of notes (from C_2 to B_7) in diapason mode, and the driving of the LCD display module.

In metronome mode, the circuit produce beat *ticks* whose sound is short and sharp, imitating that of a mechanical metronome. A useful feature is emphasising the first beat in each bar. This is done acoustically, by changing the pitch of the beat, and visually, by the lighting of a LED. An interesting feature is that the unit can provide irregular rhythms of 5/4th, 7/4th and 8/4th, as well as no rhythm — only regular beats imitating a mechanical metronome.

In diapason (tuning fork) mode, any note from C_2 to B_7 can be generated and two tuning scales can be chosen: fifths or *equal temperament*. The tone generated is quite pure with low harmonic content and is crystal controlled, for stability and better accuracy in frequency. Also, the circuit does not need any adjustment or pre-tuning: it is always on tune.

Circuit description

The heart of the circuit (see **Figure 1**) is a PIC16F84 microcontroller, from Microchip (IC1). As many of you will be aware, the PIC16F84 microcontroller is a complete computer on a chip with 1 K (words) of Flash EPROM, 68 bytes of data RAM, 64 bytes of data EEPROM, a clock circuit, 13 I/O pins, a real time clock/counter, and several other useful features. For a full description of this microcontroller and its instruction set, see the Microchip data book or look at www.microchip.com. You can download the free datasheets of this microcontroller as well as a good collection of application notes and software examples.

After developing the first version of the circuit, and after realising that there was enough free program memory in the microcontroller for

Main features

Metronome mode

- Rhythm variable between 40 and 240 beats per minute. Crystal controlled, giving high precision in time measure.
- Beats sound is short and sharp, imitating that of a mechanical metronome.
- The first beat of each measure has a different sound (higher pitch) and the led flashes.
- In the display each part of the measure is shown numbered. The numbers shift across the display to give a visual picture of the bar.
- Sound and LED can be turn on/off as desired, in independent way. Turning them on or off does not alter the internal time measurement.
- Different rhythms can be selected:
- Regular rhythms from 2 to 7 beats per bar
- Irregular rhythms of 5/4th, 7/4th and 8/4th
- No rhythm: only regular beats imitating a mechanical metronome
- All control information is permanently shown in the display.
- Headphone output and volume control

Diapason mode

- Any note, from C_2 to B_7, can be generated. Selection by octaves or incremental, note by note.
- Two tuning scales can be chosen: by fifths (F) or equal temperament (T).
- The sound generated is pure with no harmonic content. Crystal controlled for precision in frequency and stability.
- Display shows selected note and tuning scale.
- Headphones output for better quality at higher volume settings. Headphones also recommended for very low notes i.e., C_2 to B_2.

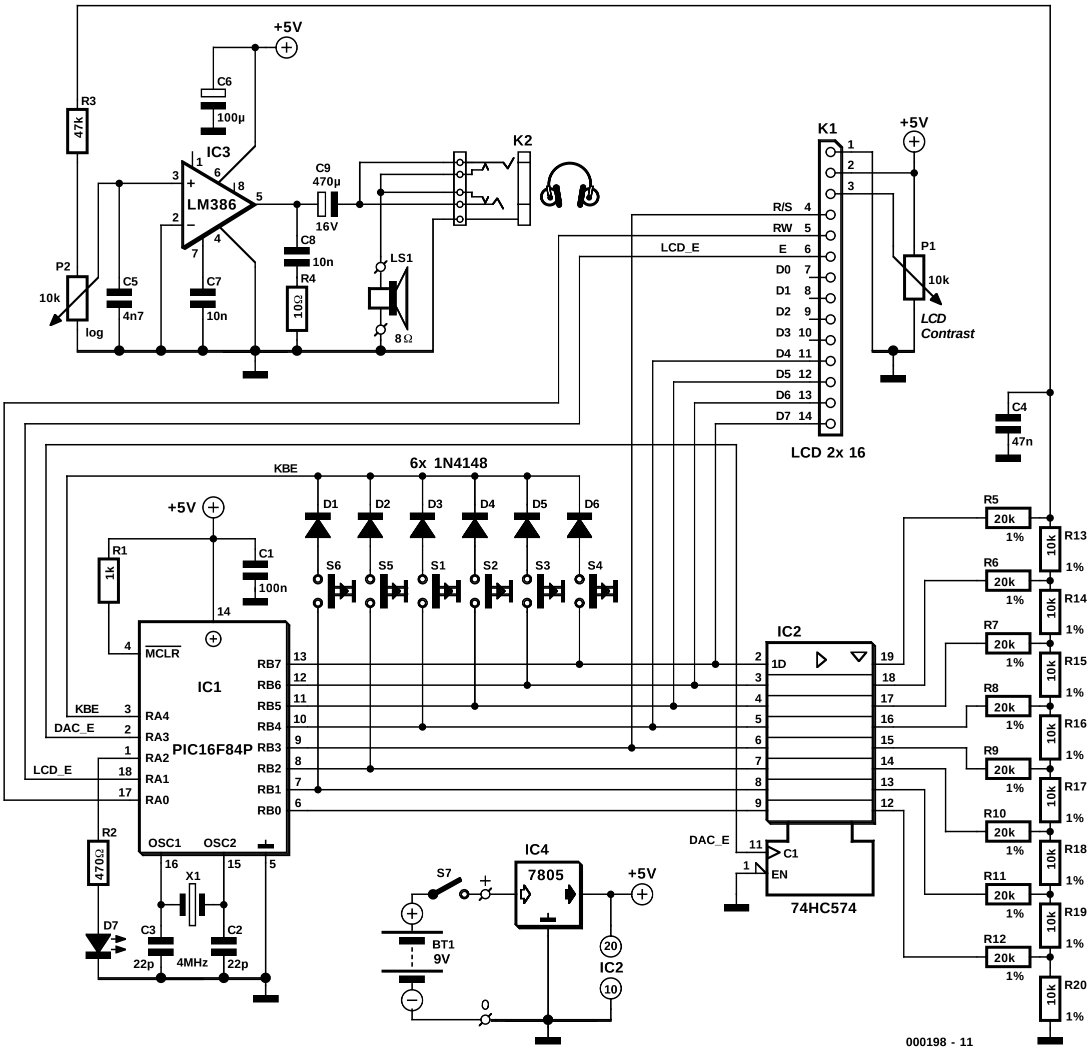


Figure 1. Circuit diagram of the Metronome/Diapason.

additional functions, it was decided to improve the quality of the sound produced by the piezoelectric sounder originally used as the sound producing device. To any musician, the *clicks* produced by a mechanical metronome are more pleasant than the short whistles produced by a piezoelectric transducer. Also, a good diapason should produce pure sinewaves, not corrupted by harmonics; this was quite far from the sound produced by the piezoelectric transducer excited by a squarewave. On the one hand, the squarewave used to excite the piezoelectric transducer was found to have a high content of odd harmonics, especially the third. On the other hand, due to the fact that piezoelectric transducers have strong peaks in their frequency response, the volume of the sound varies greatly depending of the note produced, and, for some notes, the third harmonic prevails over the fundamental. So it was decided to develop a circuit with better audio characteristics. The result is presented in this article.

Apart from the microcontroller IC1, the rest of the circuit is consists of an LCD module, six push buttons, a digital-to-analogue converter (built around IC2), and a small audio amplifier Type LM386 (IC3). The discrete DAC is built around a 74HC574 and an R-2R resistor ladder network.

One interesting thing to note is the way the bus RB0-RB7 is shared between different devices: the keyboard, the LCD unit and the DAC converter. To share the bus, the devices connected to it must be capable of being disabled when other devices use the bus. The LCD module and the DAC converter both have *enable* lines, useful for this control. For controlling the keys, each key

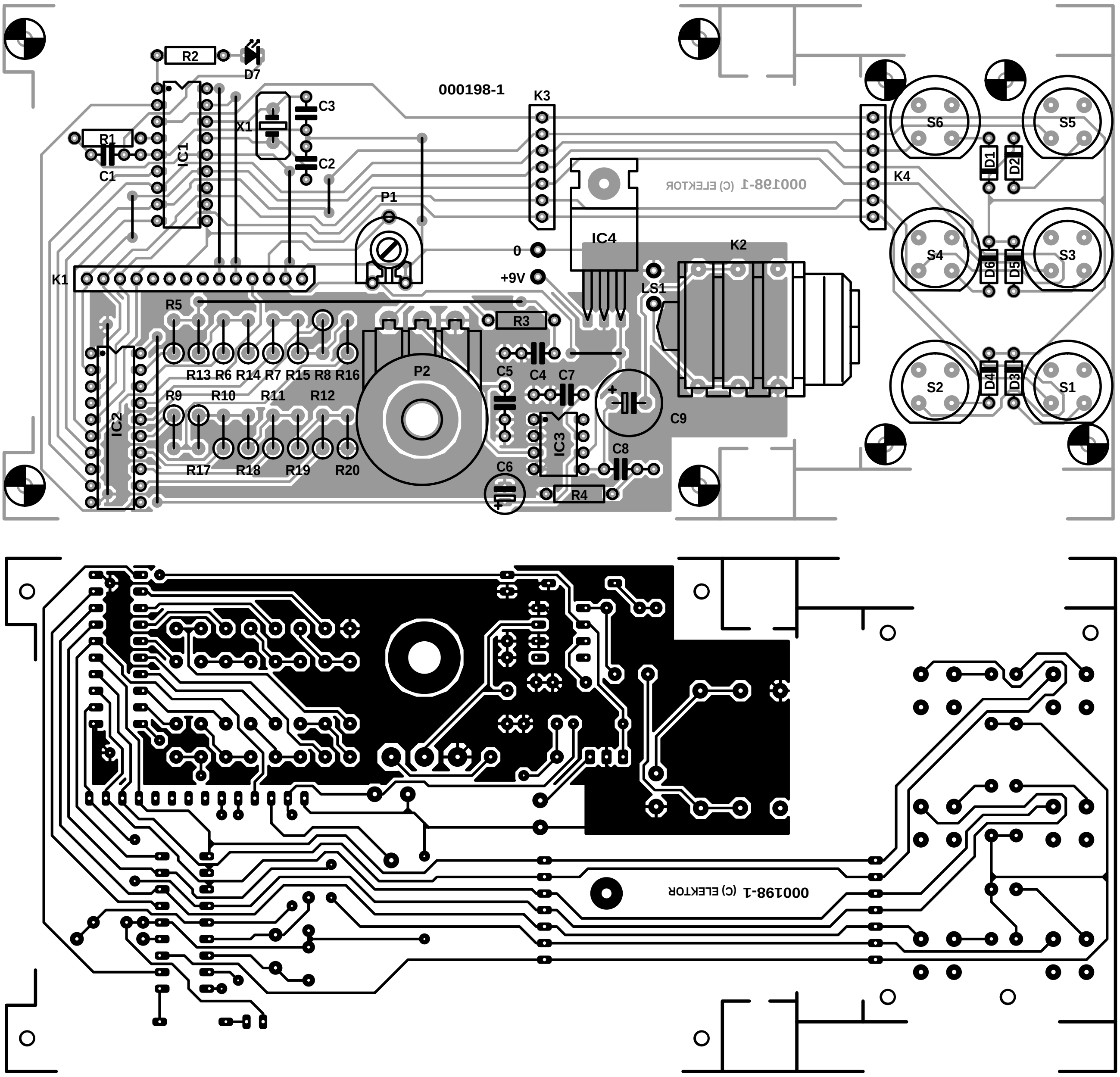


Figure 2. Copper track layout and component mounting plan of the PCB designed for the Metronome/Diapason (board available ready-made).

is connected in series with a diode (D1 through D6) to a line labelled as KBE (keyboard enable). The diodes play a double role. On one hand, if two keys are pressed simultaneously, the diodes prevent a short circuit between the corresponding I/O pins of the microcontroller. On the other hand, when the line KBE is pulled up to high level the keys are electrically isolated, avoiding interference with the other devices connected to the microcontroller bus.

An important practical observation is concerning the loudspeaker. For this design, a miniature loudspeaker is used. The volume and quality of the sound will vary considerably between one unit to another. This was noticeable not only in the intended frequency range in diapason mode but, more surprisingly, in the quality of the beats produced in metronome mode: while with some loudspeakers the sound was like a pleasant short hit, in others it was a barely audible click. So we recommend experimenting with different loudspeakers you may have available.

An optional headphones output is provided by socket K2. Useful for practising without disturbing the neighbours!

A final observation about the circuit is the low-pass filter behind the DAC. It is just a simple RC network (in fact, the combination of C4, R3, P2 and C5). As already mentioned, one design constraint was to keep the physical dimensions of the circuit as small as possible. Due to this constraint some quality in the audio part had to be sacrificed as a result of this simple filter. Nevertheless, you should take into account that when in metronome mode, to imitate the *click* sound of a mechanical unit, the loudspeaker is driven by a very

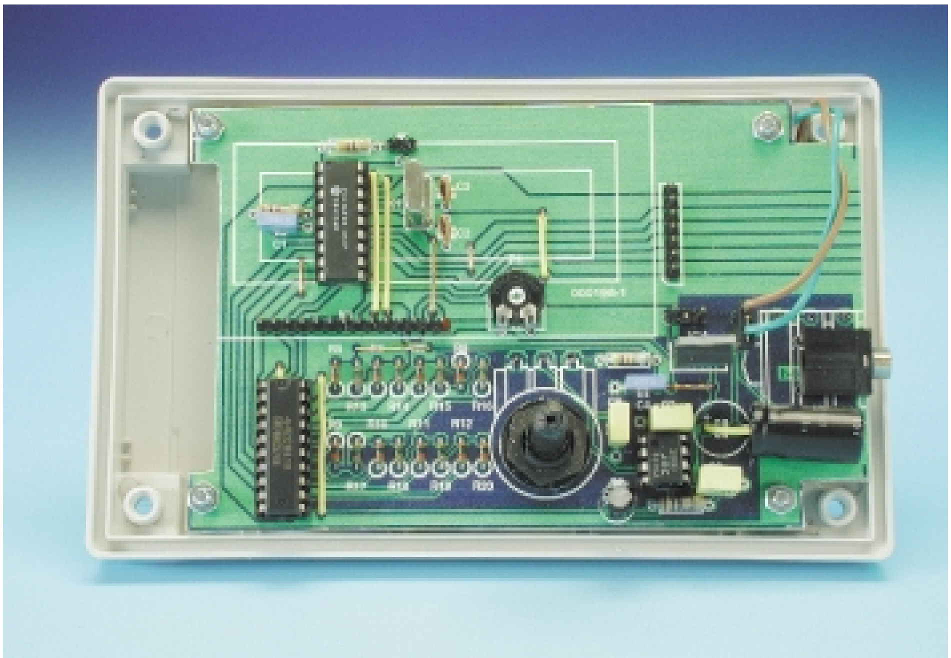
COMPONENTS LIST

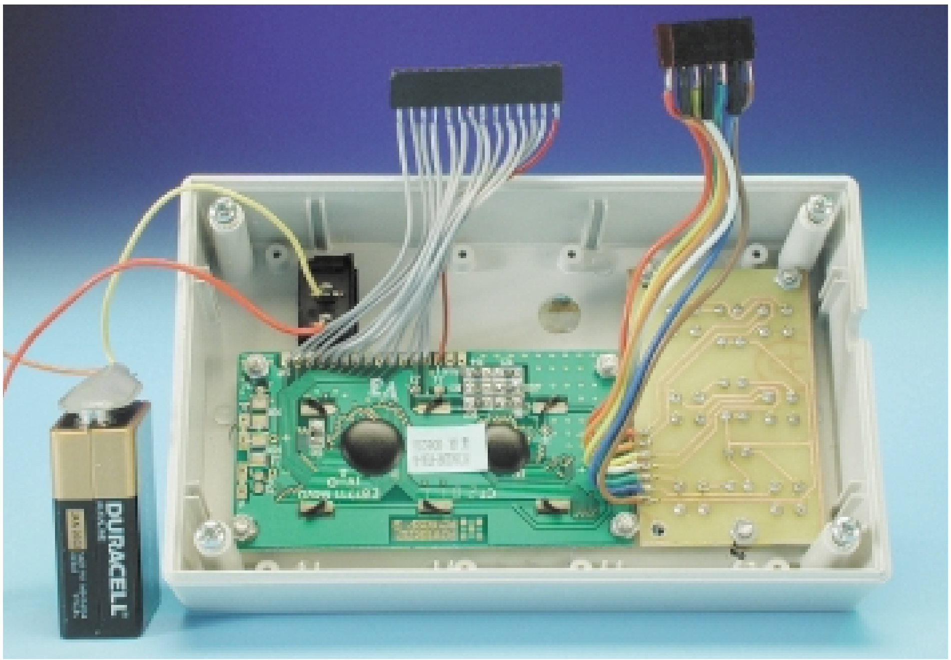
Resistors:
R1 = 1 kΩ
R2 = 470Ω
R3 = 47kΩ
R4 = 10Ω
R5-R12 = 20kΩ 1%
R13-R20 = 10kΩ 1%
P1 = 10kΩ preset
P2 = 10kΩ logarithmic potentiometer

Capacitors:
C1 = 100nF
C2,C3 = 22pF
C4 = 47nF
C5 = 4nF7
C6 = 100µF 16V radial
C7,C8 = 10nF
C9 = 470µF 16V radial

Semiconductors:
D1-D6 = 1N4148
D7 = LED, red, high efficiency
IC1 = PIC16F84-04/P, programmed, order no. **000198-41**
IC2 = 74HC574
IC3 = LM386
IC4 = 7805

Miscellaneous:
S1-S6 = pushbutton, Multimec CTL4 with cap D6K0
K2 = headphones socket, stereo, PCB mount, for 6.3mm dia. jack plug
LCD with 2 lines of 16 characters, e.g., PCF1602
Battery clip for PP3 (9V) battery
Enclosure: Retex, dim. Approx. 145 x 90 x 45mm (Conrad Electronics # 54 15 59)
PCB, order code 000189-1
Disk, all project software, order code **000189-11**





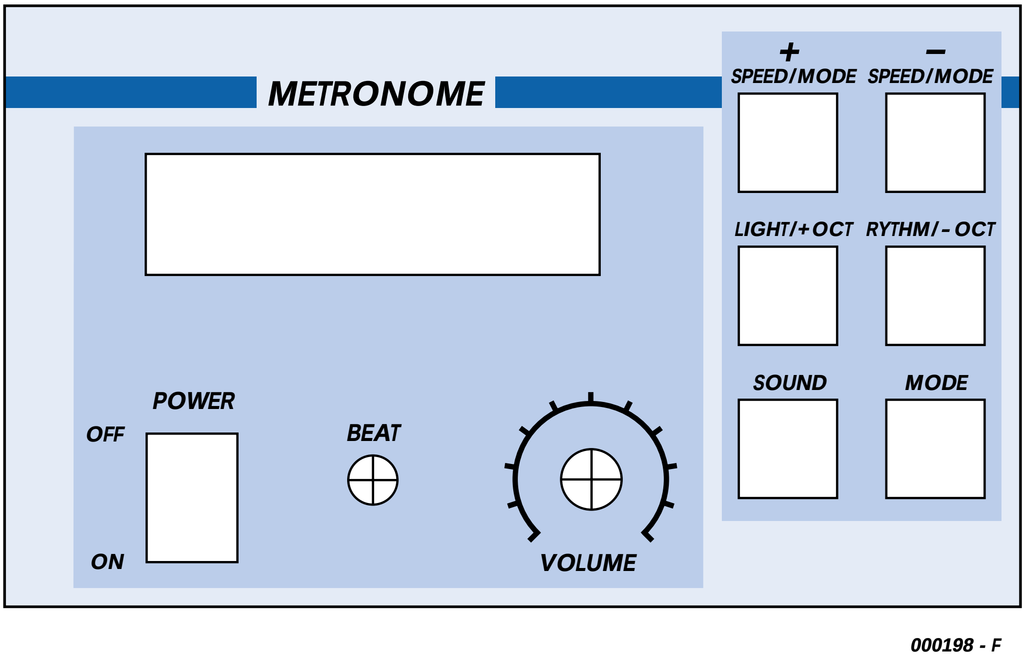


Figure 3. Suggested front panel layout at true scale.

short square wave burst (in fact, three or four cycles) to force the loudspeaker to vibrate quasi-freely, as if its cone has been hit with a stick. To get this effect it is very important that the driving signal has steep edges, which implies high harmonic content. Therefore, the cut-off frequency of the filter should be high enough to get a pleasant sound in metronome mode. Consequently, the filter was designed to strike a compromise between two opposing requirements: those of the diapason (to avoid harmonic content) and those of the metronome (to get a signal with a steep slope).

Voltage regulator IC4 reduces the battery supply voltage to +5 V.

Program description

The operation of the program that controls the PIC is quite simple. The source code is fully commented and readers should not have any problem understanding its operation. All project software is available either on diskette (Publishers' order code **000198-11**) or as a free download from our website.

Metronome mode

After initialising all the necessary data and displaying a welcome message for one second, the program enters into the main loop in metronome mode. In this mode the microcontroller must attend quasi-simultaneously to the execution of four tasks: 1) produce ticks sounds and flash the LED at relevant times; 2) update the information on the display; 3) scan the keyboard and process the function associated with key pressed, if any, and 4) update time counters. This fourth task is the most critical in metronome mode, as the main function of a metronome is to keep time count accurately. The other tasks are less critical, and a delay of a few milliseconds in attending to them is irrelevant. In this program, everything is synchronized using the timer facilities of the microcontroller. Timer0 interrupts are enabled and an interrupt is produced every 256 µs. An interrupt service routine (ISR) then takes control every exactly 256 µs and, in this routine, different time counters are incremented; also, the program checks if a tick sound is due for producing. If do, a flag is set. When the ISR routine finishes its work, control is returned to the instruction to be executed when the interrupt occurred. The other three tasks are executed in succession inside the main loop.

Diapason mode

In diapason (tuning fork) mode only two tasks need to be executed: producing a sine wave of the desired frequency and checking for key activity.

The technique used to produce a continuous sine wave is based on direct digital synthesis (DDS). In DDS, a counter (phase accumulator) is used to step through a sine look-up table, which is then fed into a digital-to-analogue converter (DAC).

In this program, the phase accumulator is a 16-bit variable called nPhase (implemented as two variables nPhaseH, nPhaseL). With each loop iteration the value in nPhaseInc is added. The variables nPhase and nPhaseInc can be regarded as having an integer portion *i* and a fractional portion *f*, i.e., *iiiiiiii.ffffffff*, where the integer portion represents steps of 1/256th cycle. This integer portion is used as an index into the sine table and the value obtained from the table is then output to the DAC.

On each loop iteration the keyboard is read to verify if any new key is pressed. If so, control is transferred to the 'read keyboard' routine, interrupting the production of the sine wave. The port B interrupt facility was not used as it is only available for four lines (RB0-RB3) and this application requires control over six keys. Although testing the keyboard on each cycle increments the loop time, the resulting sampling frequency is still high enough

(>20 kHz).

The phase increments necessary to generate sine waves of the appropriate frequencies are stored into one look-up table. This table was computed with the formula:

$$\text{Phase increment} = 256 \times f_{\text{required}} \times T_{\text{sampling}}$$

The factor 256 accounts for the effective domain of the sine function (256 steps = 2π radians). As the phase accumulator has a decimal part of 8 bits, the value 65536 is used in the formula, instead of 256. Since the sampling frequency is 25 kHz, as will be explained later, this gives a sampling period of 40 µs. In computer-speak, the formula becomes:

```
Phase increment = (int)(65536.0
   * 40.0e-6 * Frequired)
```

The phase table was generated by the program `PHASETBL.C` leaving the output in a file: `PHASETBL.EXE > PTBL.INC`. The output file `PTBL.INC` was then merged into the source file. The phase table is, in fact, two tables, as the frequencies to generate depend on the tuning mode. One table is for ‘equal temperament tuning’ mode and the other for ‘tuned by fifths’ mode.

Once the phase has been calculated, the corresponding sinewave amplitude is looked up in the sinewave table. This table contains the values (8 bits) of a sine wave in 256 steps. The table is scaled so that value range [–1.0; 1.0] is mapped to [00h; 0FFh]. In order to save space, only values for the range 0 to 90 degrees are stored — a full sine wave table would take up 256 words. The 8th bit represents the sign, so absolute values are stored as seven bits. Values are generated using the formula:

```
127*sin(phase*2*pi/256)
```

where phase is in the range 0 to 63. The sine table was generated by the program `SINETBL.C` leaving the output in a file: `SINETBL.EXE > STBL.INC`. The output file `STBL.INC` was then merged into the source file.

Returning to program operation, when the program enters into diapason mode all interrupts are disabled and, after some initialisations, the program enters a loop. In this loop four tasks are executed in sequence:
- update the sine wave phase. This takes 6 µs;
- calculate the sample value by looking up in the sinewave table (14 µs);
- send the sample value to the DAC (5 µs); and finally
- verify if any key is pressed and if not, repeat the loop (15 µs).

The total loop time is always fixed, unless a key is pressed, and it takes 40 µs. This implies that the sampling frequency is 25 kHz, high enough for the maximum sound frequency to produce (B_7, nearly 8 kHz) whilst obviating the need for strong low-pass filtering after the DAC.

When a keyboard activity is detected, the sinewave production is suspended and the key function is processed.

Construction

The PCB artwork for the project is given in **Figure 2**. Construction of the metronome is fairly easy as no adjustment is necessary apart from the desired contrast of the LCD. Depending on the enclosure you intend to use, the keyboard section may or may not be separated from the main board. If you employ miniature resistors, these may be mounted horizontally rather than vertically as indicated by the component overlay.

Check the orientation of all polarised devices, particularly the diodes connected to the keys. For the display, any general-purpose 2-line, 16-character display having the Hitachi HD44780 chip should be suitable. The HD44780 is widely used between display manufacturers. If the LCD you employ has a backlight, then this should be connected extra. No provision has been made for adjusting the frequency of the crystal oscillator, as any error should be negligible. Nevertheless, should you like to adjust it, simply substitute one of the capacitors C2 or C3 by a small trimmer of 40 pF.

A suggested fascia for the Metronome/Diapason is shown in **Figure 3** (not available ready-made).

Operation

When the unit is switched on, the welcome message is displayed for about one second. Next, it goes into Metronome mode with ternary rhythm and a tempo of 120 beats per minute. The user can then change the settings as desired by pressing the keys. The key functions should be easy to learn once used a couple of times. Some labels on the keys have two names: the first one is applied when in metronome mode and the second when in Diapason mode. The functions in Metronome and Diapason mode are explained in **Table 1** and **Table 2** respectively.

(000198-1)

Table 1. Functions available in Metronome mode

KEY	FUNCTION
+	Increment tempo (bpm). Maximum is 240 bpm.
–	Decrement tempo. Minimum is 40 bpm.
Light/ –Oct	Switch LED on / off.
Rhythm/ +Oct	Cycle through different rhythms: 1, 2, 3, 4, 5, 6, 7, 5/4th, 7/4th, 8/4th.
Sound	Switch the sound on / off.
Mode	Change to Diapason mode, equal temperament tuning.

Table 2. Functions available in Diapason (tuning fork) mode

KEY	FUNCTION
+	Increment note. Highest is B7.
–	Decrement note. Lowest is C2.
Light/ –Oct	Decrement octave, e.g., if current note is G4, change to G3.
Rhythm/ +	Oct Increment octave.
Sound	Switch sound on / off
Mode	From equal temperament tuning, change to tuning by fifths. From tuning by fifths mode, change to Metronome mode.